

## **REMARKS**

The claims are claims 1, 2, 4, 5 and 7 to 11.

Claims 1, 4 and 7 are amended. Claims 3 and 6 are canceled. New claims 8 to 11 are added. Claims 1 and 4 are amended to include the reduced power limitations of respective canceled claims 3 and 6. Claim 1 is further amended in response to the Examiner's claim objections. Claim 7 is amended to recite that instructions are statically scheduled by a compiler. Claims 8 and 10 recite the reduced power operation occurs by not fetching at least one operand and not toggling the corresponding register read port of the functional unit. The application states at page 3, lines 11 to 14 that reading register operands wastes energy for predicated instructions that are not committed and states at page 3, line 19 to 21 that this invention saves power in these wasted instructions. The application states at page 25, lines 11 to 16 that maintaining the functional units register read ports reduces power consumed. Those skilled in the art would understand that not fetching operands prevents toggling on the functional unit register read ports. Claims 9 and 11 recite not powering the functional unit during the execution pipeline phase as taught in this application at page 26, lines 12 and 13.

Claims 1, 2, 3, 4, 5, 6, and 7 were rejected under 35 U.S.C. 102(e) as being anticipated by Kling et al U.S. Patent No. 6,883,089.

Claims 1 and 4 recite subject matter not anticipated by Kling et al. Claim 1 recites "each functional unit is further operative responsive to a predicate instruction during said instruction decode pipeline phase to nullify said predicate instruction of a following execution phase by operating at a reduced power state relative to normal instruction operation if said predicate register has said second state and said corresponding scoreboard bit has

said second state." Claim 4 similarly recites "nullifying a predicate instruction for a following execution phase by operating said corresponding functional unit at a reduced power state relative to normal instruction operation if said corresponding predicate register has said second state and said corresponding scoreboard bit has said second state during a prior decode phase." This subject matter was originally recited in claims 3 and 6. Regarding claim 3 the OFFICE ACTION states at page 6, lines 7 to 10:

*"Note that when instructions are discarded, bubbles are sent through the processor. These bubbles generally take less power than instructions. Further, a reduced power state is a relative term. As such, the processor is always running at a reduced power state compared to something."*

This statement in the OFFICE ACTION is argument and not evidence of anticipation. Firstly, Kling et al fails to disclose the "bubbles" noted by the Examiner. Secondly, there is no citation to any evidence in Kling et al that such "bubbles" take less power than instructions. Claims 1 and 4 now recite reduced power "relative to normal instruction operation" thus tying this reduction to a known quantity. Thus while the processor may always be "running at a reduced power state compared to something" it will not always be running at reduced power relative to normal instruction operation as recited in claims 1 and 4. Kling et al states at column 3, lines 38 to 41:

*"After execution, the result is written back to the register file 168, if the predicate is known and 'true'. The result is discarded if the predicate is known and 'false'. If the predicate is still unresolved the result is written back to a temporary result buffer 172 (associative buffer)."*

This teaching of Kling et al implies that the result is discarded if the predicate is false "After execution." Thus this implies normal instruction operation during the execution phase contrary to the language of claims 1 and 4 quoted above. Accordingly, claims 1 and 4 are allowable over Kling et al.

Claim 4 recites further subject matter not anticipated by Kling et al. Claim 4 recites three alternative results of instruction operation dependent upon the availability and state of the predicate register data. These are: performing a data processing operation "if said corresponding predicate data register has a first state"; nullifying a predicate instruction by not writing the result "if said corresponding predicate register has a second state opposite to said first state"; and nullifying a predicate instruction by operating at a reduced power state "if said corresponding predicate register has said second state and said corresponding scoreboard bit has said second state during a prior decode phase." The above quoted portion of Kling et al (within the portion cited in the OFFICE ACTION as anticipating the nullifying instruction limitation in claim 1) teaches the first and second of these limitations of claim 4. The final limitation of operating at reduced power is a different limitation than discarding the result as taught in Kling et al and the second limitation of claim 4 not writing the result. Thus this last limitation of claim 4 differs from and is not anticipated by Kling et al.

Claims 2 and 5 recite subject matter not anticipated by Kling et al. Claims 2 and 5 recite resetting "a scoreboard bit to a second digital state upon nullification of said instruction designating said corresponding data register as a destination operand data register." Regarding claim 3 the OFFICE ACTION states at page 6, lines 1 to 3:

*"Note that the processor must update the scoreboard in response to nullifying an instruction. If the scoreboard is not updated, the processor could stall indefinitely waiting for operands to become available."*

The Applicants respectfully submit that the Examiner's statement "If the scoreboard is not updated, the processor could stall indefinitely waiting for operands to become available" is incorrect and not what is recited in claims 2 and 5. Firstly, claims 2 and 5 do not state that operation stalls in the scoreboard is not updated. The instruction subject to nullification in claims 2 and 5 is "said instruction designating said corresponding data register as a destination operand." Base claims 1 and 4 recite this "corresponding data register" is the predicate data register which conditions the dependent instruction. Thus this is an instruction that writes to the predicate register and not the dependent instruction. Note further if this scoreboard register is not reset, claim 2 does not recite that the dependent instruction stalls. Claim 5 specifically recites a condition in the "performing a data processing operation" paragraph under which the dependent instruction executes without reference to the state of the scoreboard bit. As taught in the application at page 23, line 13 to page 6, line 8, the scoreboard bit is set upon detection of a write to the corresponding predicate register (page 23, lines 15 to 21) and reset upon either a commit of that write (page 23, lines 24 to 29) or a nullification of that write (page 23, line 29 to page 24, line 5). This application teaches that only an early nullification of the dependent instruction depends on the scoreboard bit. This application teaches at page 25, line 20 to page 26, line 4 that if the early nullification decision cannot be made the instruction proceeds with the regular nullification decision. Base claim 1 of claim 2 does not recite this regular nullification decision but does not exclude it. Base claim 4 of

claim 5 recites this regular nullification decision which does not depend upon the status of the scoreboard bit. Accordingly, claims 2 and 5 are allowable over Kling et al.

Claim 7 recites subject matter not anticipated by Kling et al. Claim 7 recites "statically scheduling instruction execution via a compiler." Kling et al clearly teaches dynamic instruction scheduling, that is instruction scheduling that occurs at run-time and not at compile time. Thus the teachings of Kling et al do not anticipate this limitation. Accordingly, claim 7 is allowable over Kling et al.

Claim 7 recites additional subject matter not anticipated by Kling et al. Claim 7 recites "scheduling via said compiler a last write to a data register a decode phase of a predicate instruction designating said data register as a predicate register." Kling et al teaches that the data within the predicate data must be available and valid before the execution phase of the predicated instruction. Kling et al fails to teach that any change to the predicate data register must be scheduled before the decode phase of the predicate instruction. Accordingly, claim 7 is allowable over Kling et al.

New claims 8 to 11 recite subject matter not anticipated by Kling et al. New claims 8 and 10 recite achieving the reduced state by "not fetching at least one instruction operand and not toggling a correspond register read port during said following execution phase." The Applicants respectfully submit that Kling et al teaches neither nor fetching operands nor not toggling register read ports. New claims 9 and 11 recite achieving the reduced power state by "not powering said functional unit during said following execution phase." The Applicants respectfully submit that Kling et al fails to teach not powering a functional unit during an execution phase. Accordingly, claims 8 to 11 are allowable over Kling et al.

The Applicants respectfully submit that all the present claims are allowable for the reasons set forth above. Therefore early reconsideration and advance to issue are respectfully requested.

If the Examiner has any questions or other correspondence regarding this application, Applicants request that the Examiner contact Applicants' attorney at the below listed telephone number and address to facilitate prosecution.

Texas Instruments Incorporated  
P.O. Box 655474 M/S 3999  
Dallas, Texas 75265  
(972) 917-5290  
Fax: (972) 917-4418

Respectfully submitted,

/Robert D. Marshall, Jr./  
Robert D. Marshall, Jr.  
Reg. No. 28,527